IoT Security Validation

Joshua French, Vincent Johnson, Jordan McKillip, Marcus Reecy

Megan Ryan

Kristin Rozier

Project Vision

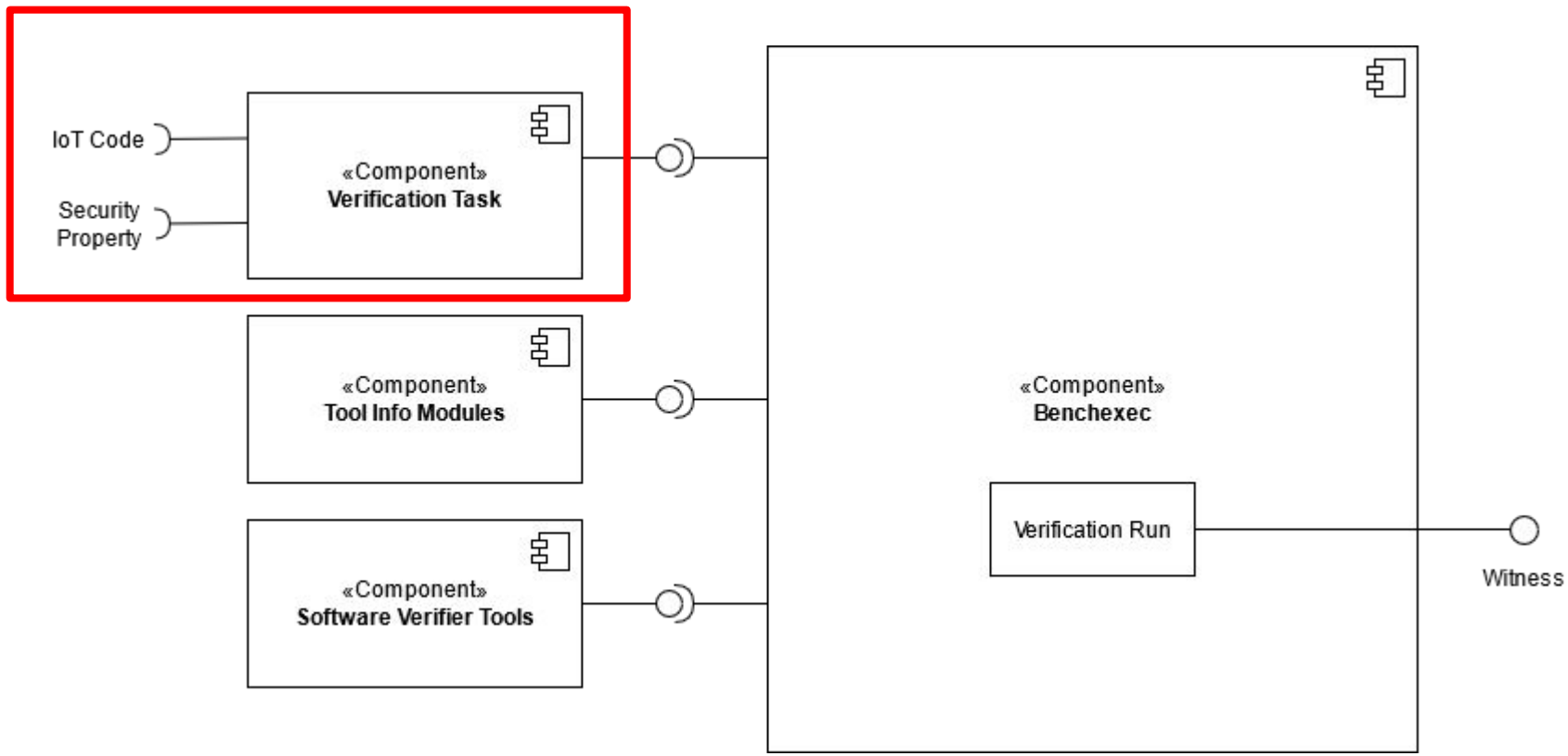
The world of Internet of Things (IoT) grows every day into more and more private aspects of our lives. Securing these devices is incredibly important and our project contributes to doing exactly that.

IoT: The network of physical devices that are embedded with sensors, software and other pieces of hardware for the purpose of connecting and exchanging data with other devices and systems over the internet.

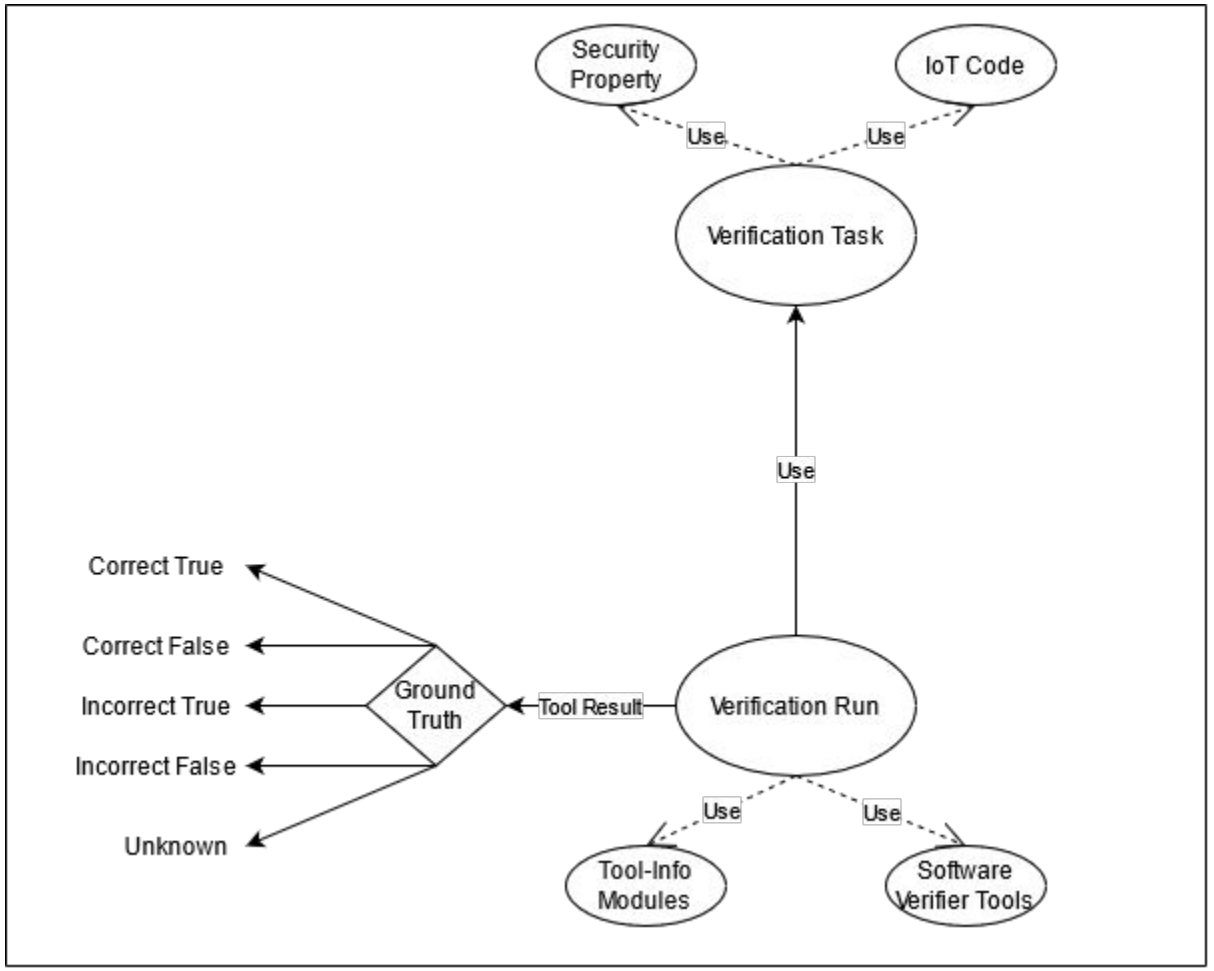
Project (SV-COMP)

- SV-COMP (Software Verification Competition)
  - A competition aimed at driving invitation for methods, technologies and tools that verify software
  - Not focused on IoT/Security, but tools are applicable
- Benchmarks: Code that is built to be tested and verified with different verification tools
- Verification Tools: Programs that are designed to test the different security properties and determine the security of a given application

System Design



System Design - Use Case Diagram



CWEs

- Common Weakness Enumeration: A category of system for software weaknesses and vulnerabilities. Part of a wider goal to understanding flaws in software

- Memory Safety
  - CWE-119: Improper Restrictions of Operations within the Bounds of a Memory Buffer
- Reach Safety
  - CWE-200: Exposure of sensitive information to an unauthorized Actor
- No Overflow
  - CWE-121: Stack-based buffer overflow

Engineering Requirements

- Functional
  - Each tool can:
  - Report its own version
  - Report result of each security property
  - Can be implemented and compiled using the SV-COMP standard computer
- Non-functional
  - Can be downloaded, replicated, and evaluated
  - Can be archived in a ZIP file
  - File packages should contain all required libraries
  - File packages should contain no unnecessary data, code, etc.
- Technical and/or other constraints
  - Written in C
  - IoT libraries modified such that the tool can assess properties.

Engineering Constraints

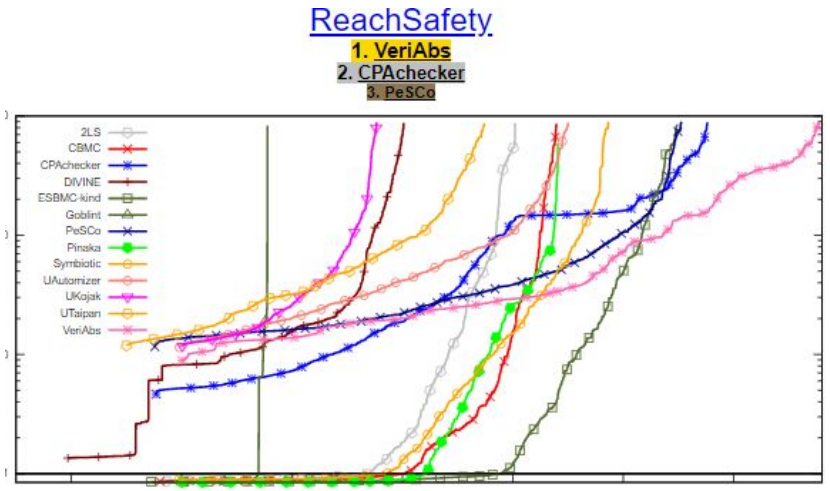
- Utilization of the tools requires a lot of time
  - Average time for a single run could take up to 15 minutes.
- Size of files paramount - larger files take longer to run
- To run benchexec and tools:
  - Ubuntu 18.04, with at minimum, a memory limit of 15 GB (14.6 GiB) of RAM and a limit to 8 processing units of a CPU conforming to standard SV-COMP guidelines
- Required software:
  - GCC, make, python3, Benchexec, SV-COMP tools

Selecting Security Properties

- The original security properties chosen based on the following:
  - Far reaching, exploitive, and examples have documentation (Within CWE)
- No overflow: Protecting against overflows to write into restricted code(Signed ints and their overflow that cannot be allowed to be present within a program)
- Reach safety: Checking to see if function calls are reachable within the program
- Memory safety: A set of memory safety property that is a conjunction of valid memory cleanup, valid-deref, valid free, and valid memtrack.

Selecting Software Verification Tools

- Tools were chosen as the recorded best to test these particular properties
  - Influenced by 2021 SV-COMP
- Predator HP: Best for Memory Safety category
- VeriAbs: Best for Reach Safety category
- CPAChecker: Best for No-Overflow category



Creating a Verification Task

- Choose benchmark code (library or device code)
  - Some benchmarks we created entailed using the entire library, while others entailed using only a few methods.
  - Refactor code to fit run by itself (without external libraries) and replace inputs with model checker input:

```
extern uint8_t __VERIFIER_nondet_uint8_t(void);
extern uint16_t __VERIFIER_nondet_uint16_t(void);
extern int16_t __VERIFIER_nondet_int16_t(void);
extern uint32_t __VERIFIER_nondet_uint32_t(void);
extern int32_t __VERIFIER_nondet_int32_t(void);
extern float __VERIFIER_nondet_float(void);
```

Supporting Files

- Import benchmarks into benchmark directory (with license information)
- Create a .yml file for the benchmark

```
format_version: '2.0'
input files: '<filename>.i'

properties:
property_file: ../properties/<propertyfile>.prp
expected_verdict: <True / False>

options:
language: C
data_model: ILP32
```

- Create README / Makefile in directory (Required to run in SV-COMP)
- Create a .set file containing the below to specify which benchmarks to run

```
<benchmark-dir>/*.yml
```

Compiling the Benchmark

- Compile the benchmark files with Benchexec exceptions using make
  - This is the stage may contain errors you need to fix
- Create .i files
  - gcc -E <.c file> -o <.i file> -P -m64
- Verify the integrity of the benchmarks using python script
  - python check.py

Using the Benchmark

- Download the archived tools from SV-COMP and unzip in directory
- In ../bench-defs/benchmark-defs/<tool>.xml
  - Check to make sure hardware specifications are the same for your machine
  - Create run definitions to run your benchmarks:

```
<rundefinition name="sdmay21-41_no-overflow">
  <tasks name="No-Overflow-Sprint1">
    <includesfile>../sv-benchmarks/c/sdmay-21-41.set</includesfile>
    <propertyfile>../sv-benchmarks/c/properties/no-overflow.prp</propertyfile>
  </tasks>
  </rundefinition>
```

- Return to unzipped tool directory and run:

  ../bench-defs/benchmark-defs/<tool>.xml -r <run-definition-name>

Verification Run: LightBlueLatch_Sensor

T/F = Correct True/False

IoT BENCHMARKS	CPA-SEQ			
	No Overflows	Mem Safety	Reach Safety	Mem Cleanup
LightBlueLatch_Sensor_patched_true-no-overflow_true-mem-safety.c	T	T	x	x
LightBlueLatch_Sensor_vulnerable_false-no-overflow_false-mem-safety	F	F	x	x

IT/IF = Incorrect True/False | U = Unknown | ? = Tool Error

PredatorHP				VeriAbs**			
No Overflows	Mem Safety	Reach Safety	Mem Cleanup	No Overflows	Mem Safety	Reach Safety	Mem Cleanup
T	U	x	x	T	T	x	x
U	U	x	x	F	F	x	x

```c
extern uint8_t __VERIFIER_nondet_uint8_t(void);
extern uint16_t __VERIFIER_nondet_uint16_t(void);
extern int16_t __VERIFIER_nondet_int16_t(void);
extern uint32_t __VERIFIER_nondet_uint32_t(void);
extern int32_t __VERIFIER_nondet_int32_t(void);
extern float __VERIFIER_nondet_float(void);

const char keycode[] = {13, 14, 15, 16};

int main() {
		while(1) {
			loop();
		}
}

void loop(){
  char buffer[10];
  size_t length = 10;
  static char last_value = 0;
  static unsigned char index = 0;
  static char lock_state = 0;
  int i = 0;
  for (i; i <10; i++) {
	buffer[i] =  __VERIFIER_nondet_uint8_t();
  }

  if( length > 0 ){
    if(buffer[0] != last_value){  // Check to see if
	if(buffer[0] == keycode[index]){
	  index++;
	  if(index == KEYCODE_SIZE){
	  index = 0;
	  }
	}else{
	  index = 0;
	}
    }
    last_value = buffer[0];
  }

}
```

Verification Run: MQTT_num_rem_len_bytes

IoT BENCHMARKS	CPA-SEQ			
	No Overflow s	Mem Safety	Reach Safety	Mem Cleanup
mqtt_num_rem_len_bytes_patched.c	*x*	*T*	*x*	*T*
mqtt_num_rem_len_bytes_vulnerable.c	*x*	*F*	*x*	*T*

IT/IF = Incorrect True/False U = Unknown ? = Tool Error

PredatorHP				VeriAbs**			
No Overflow s	Mem Safety	Reach Safety	Mem Cleanup	No Overflow s	Mem Safety	Reach Safety	Mem Cleanup
x	T	*x*	*U*	*x*	*T*	*x*	*T*
x	F	*x*	*U*	*x*	*T*	*x*	*T*

```c
extern uint8_t __VERIFIER_nondet_uint8_t(void);
extern uint16_t __VERIFIER_nondet_uint16_t(void);
extern int16_t __VERIFIER_nondet_int16_t(void);
extern uint32_t __VERIFIER_nondet_uint32_t(void);
extern int32_t __VERIFIER_nondet_int32_t(void);
extern float __VERIFIER_nondet_float(void);

uint8_t mqtt_num_rem_len_bytes(const uint8_t* buf);

int main() {
        while (1) {
                const uint8_t* buf = (void*) __VERIFIER_nondet_uint8_t();
                mqtt_num_rem_len_bytes(buf);
        }
}

uint8_t mqtt_num_rem_len_bytes(const uint8_t* buf) {
        uint8_t num_bytes = 1;

        //printf("mqtt_num_rem_len_bytes\n");

        if ((buf[1] & 0x80) == 0x80) {
                num_bytes++;
                if ((buf[2] & 0x80) == 0x80) {
                        num_bytes ++;
                        if ((buf[3] & 0x80) == 0x80) {
                                num_bytes ++;
                        }
                }
        }
        return num_bytes;
}
```

Project Statistics

- 36 benchmarks produced
  - 18 Considered “Patched” - (No errors expected)
  - 18 Considered “Vulnerable” - (Property errors expected)
- Each tested with 4 properties
- Tested with 3 different tools

Coding Results

```
executing run set 'sdmay21-41_no-overflow.ReachSafety-no-overflow'     (3 files)
03:12:19   LightBlueLatch_sensor/LightBlueLatch_Sandbox_Keycode.yml    true                1.05    0.79
03:12:20   Particle_Pi_Camera/Particle_Pi_Camera.yml                   unknown             0.58    0.54
03:12:21   Azure_Plug_Play_Bridge/SerialPnP.yml                        TIMEOUT           900.51  450.30

executing run set 'sdmay21-41_reach-safety.ReachSafety-unreach-call'     (2 files)
03:19:53   LightBlueLatch_sensor/LightBlueLatch_Sandbox_Keycode.yml    UNKNOWN             0.08    0.29
03:19:53   Azure_Plug_Play_Bridge/SerialPnP.yml                        UNKNOWN             0.07    0.29

executing run set 'sdmay21-41_mem-safety.ReachSafety-memsafety'     (3 files)
03:19:54   LightBlueLatch_sensor/LightBlueLatch_Sandbox_Keycode.yml    true                0.87    0.29
03:19:55   Particle_Pi_Camera/Particle_Pi_Camera.yml                   unknown             0.57    0.54
03:19:56   Azure_Plug_Play_Bridge/SerialPnP.yml                        TIMEOUT           900.50  450.41


executing run set 'sdmay21-41_no-overflow.ReachSafety-no-overflow'     (3 files)
20:26:07   LightBlueLatch_sensor/LightBlueLatch_Sandbox_Keycode.yml    true               16.38    6.95
20:26:15   Particle_Pi_Camera/Particle_Pi_Camera.yml                   true               20.84    8.34
20:26:23   Azure_Plug_Play_Bridge/SerialPnP.yml                        true               18.48    7.60

executing run set 'sdmay21-41_reach-safety.ReachSafety-unreach-call'     (2 files)
20:26:31   LightBlueLatch_sensor/LightBlueLatch_Sandbox_Keycode.yml    true               15.93    6.40
20:26:38   Azure_Plug_Play_Bridge/SerialPnP.yml                        true               19.41    7.65

executing run set 'sdmay21-41_mem-safety.ReachSafety-memsafety'     (3 files)
20:26:46   LightBlueLatch_sensor/LightBlueLatch_Sandbox_Keycode.yml    true               16.16    6.22
20:26:53   Particle_Pi_Camera/Particle_Pi_Camera.yml                   true               21.19    8.32
20:27:02   Azure_Plug_Play_Bridge/SerialPnP.yml                        true               19.04    7.58
```

Technical Challenges

- Getting SV-COMP / tools running
  - VM limitations required constant testing
- Analyzing libraries / creating benchmarks from them
- Making benchmarks runnable
  - Had to have them meet SV-COMP standards
- Creating our own properties

```
fnet_ip6_multicast_list_entry_t *_fnet_ip6_multicast_find_entry(fnet_netif_t *netif, const fnet_ip6_addr_t *group_addr )
{
    fnet_index_t                    i;
    fnet_ip6_multicast_list_entry_t *result = FNET_NULL;

    /* Find existing entry or free one.*/
    for(i = 0u; i < FNET_CFG_MULTICAST_MAX; i++)
    {
        if((fnet_ip6_multicast_list[i].user_counter > 0u)
           && (fnet_ip6_multicast_list[i].netif == netif)
           && FNET_IP6_ADDR_EQUAL(&fnet_ip6_multicast_list[i].group_addr, group_addr))
        {
            result = &fnet_ip6_multicast_list[i];
            break; /* Found.*/
        }
    }

    return result;
}
```

Technical Challenges cont.

- Some tools are more successful at testing certain properties than others
- Some benchmarks may error out with some tools

CPA-SEQ			PredatorHP			VeriAbs		
No Overflows	Mem Safety	Reach Safety	No Overflows	Mem Safety	Reach Safety	No Overflows	Mem Safety	Reach Safety
T	T	T	T	T	?	T	T	T
T	U	T	U	U	?	T	T	T
T	U	?	U	U	?	T	T	T
T	*U*	*T*	*?*	*?*	*?*	*U*	*U*	*T*
T	*T*	*T*	*?*	*U*	*?*	*?*	*T*	*T*
T	*T*	*?*	*?*	*?*	*?*	*U*	*T*	*T*
T	*U*	*T*	*?*	*U*	*?*	*T*	*U*	*T*
T	*T*	*T*	*?*	*U*	*?*	*?*	*T*	*T*
T	*U*	*?*	*?*	*?*	*?*	*U*	*U*	*T*
T	*U*	*T*	*?*	*?*	*?*	*T*	*T*	*T*
T	*U*	*T*	*?*	*U*	*?*	*T*	*T*	*T*
						?	*?*	*?*
T	T	T	T	T	T	T	T	T
T	T	T	T	T	T	T	T	T
T	T	T	U	U	U	T	T	T
U	IF	U	T	T	T	T	T	T
T	T	T	T	T	T	T	T	T
U	T	T	U	U	U	T	T	T

Conclusions

- IoT devices are some of the most insecure devices on the market right now. Our project, along with SV-COMP, gives the opportunity to verify that the device software you are using is not vulnerable to any software related issues.
- We are currently working on getting our project and benchmarks submitted to SV-COMP for potential use in their competitions.
- We hope our project contributions may lead to future additions to SV-COMP to make it more relevant to IoT security.

Team Contributions

- Jordan McKillip
  - Developer/Website
- Joshua French
  - Developer
- Vincent Johnson
  - Project Lead/Developer
- Marcus Reecy
  - Developer/Website